PREPRINT ANL/MCS-P740-0199, JANUARY, 1999, MATHEMATICS AND COMPUTER SCIENCE DIVISION, ARGONNE NATIONAL LABORATORY, ARGONNE, IL 60439, USA.

Advanced Computational Techniques for Laue Diffraction Analysis*

Zhong Ren,[†] Rongqin Sheng,[‡] and Stephen J. Wright[§]

February 4, 1999

Abstract

We describe LaueView, a code for processing the measured intensity data in Laue X-ray diffraction experiments to obtain corrected structure amplitudes for each reflection that take account of the various distortion effects. The resulting corrected intensity data can then be used to recover the molecular structure by isomorphous refinement or by solution of the phase problem. We describe the key numerical techniques used in LaueView and outline the improvements we made to obtain a new, more efficient, and parallel version of the code. We conclude with some computational results obtained on a real data set that illustrate our improvements. The basic principles of the Laue method are described in an appendix, where we outline the distortions in the measured intensity data due to effects such as blurring, overlap of the spots, the nonuniform distribution of intensities in the incident X-ray beam, and absorption effects of various types.

1 Introduction

X-ray diffraction analysis of crystals reveals the structure of the molecules that make up the crystal. The problem of determining the structure of large molecules, particularly biomolecules, is an important one that tests the limits of current computational capabilities.

When X-rays are beamed onto a crystal, they are diffracted to produce a regular array of "spots" of varying intensity on an area detector. The locations of the spots are determined by the crystal lattice, while their intensities depend on the spatial and temporal average conformation of all molecules in the crystal and during data collection. In this paper, we focus on the important process of *scaling* the measured intensities to obtain the structure factor amplitudes, and thus correcting for various effects embedded in the raw data. We also identify other issues that arise in the processing of the measured intensity data, such as prediction of the diffraction pattern and determination of the intensity of each spot on the detector by integrating over a finite area of irregular shape. Processing of

*This work was supported by the Mathematical, Information, and Computational Sciences Division subprogram of the Office of Computational and Technology Research, U.S. Department of Energy, under Contract W-31-109-Eng-38, by a DOE Grand Challenge Application Grant, B&R No. KJ-01-01-03-0, and in part by NIH Grant RR07707 to Keith Moffat.

†Department of Biochemistry and Molecular Biology, The University of Chicago, 920 East 58th Street, Chicago, IL 60637, USA; renz@cars.uchicago.edu

‡Mathematics and Computer Science Division, Argonne National Laboratory, 9700 South Cass Avenue, Argonne, IL 60439, USA; sheng@mcs.anl.gov

§Mathematics and Computer Science Division, Argonne National Laboratory, 9700 South Cass Avenue, Argonne, IL 60439, USA; wright@mcs.anl.gov

the diffraction data is a challenging task from many viewpoints: modeling and analysis, numerical techniques, and computational requirements.

We are interested in the Laue method of X-ray diffraction, in which the incident X-ray is not monochromatic, but rather is made up of a spread of different wavelengths. The Laue technique produces complex diffraction images containing many spots, some of which overlap or are superimposed. Monochromatic (single-wavelength) beams are more widely used, and they produce much simpler diffraction images that are more easily analyzed. Despite the additional complications, however, the Laue technique is more suitable in situations in which data must be gathered quickly, such as when our intention is to observe reactions in progress (see, for example, Genick et al. [2]). Another major advantage is that Laue diffraction is more suited to synchotron sources (such as the Advanced Photon Source at Argonne National Laboratory), which naturally produce bright X-rays with a spread of wavelengths. Moffatt [4] discusses the Laue technique, while Clifton et al. [1] describe the evaluation of Laue diffraction images.

Figure 1 shows a representative Laue diffraction image collected from a restrictocin crystal. This crystal is in space group $P2_1$ with cell constants $a = 50.2$Å, $b = 82.2$Å, and $c = 38.0$Å; and $\beta = 100.5°$. This crystal has a low mosaic spread and diffracts robustly to a high resolution of 1.5Å. Figure 2 is a Laue image of a thermal stable β-glucosidase in space group $P42_12$ with cell constants $a = b = 220$Å and $c = 100$Å. This crystal has much higher mosaic spread, revealed by the elongated diffraction spots. A complete static data set requires tens of such images depending on the crystal space group, resolution and wavelength bandpass of the beamline used. A time-resolved data set that imaged a reaction in progress would require a data volume equivalent to 100 static data sets, that is, thousands of raw images. The exposure time available to collect those images is very short, typically subseconds. On the other hand, the elapsed time of such experiment can range from hours to days, most of which is spent on detector readout and crystal manipulation. Months are typically required for crystallographers to process the images acquired from a synchrotron run. It is this aspect of the experimental process that we discuss in this paper.

To process intensity data from a Laue diffraction experiment, the measured spot intensities must be adjusted to account for effects of variation in intensities of the incident beam as a function of wavelength, energy absorption by the crystal, polarization of the X-ray beam, temperature effects, and so on. In addition, harmonic overlapping spots must be "deconvoluted" and a specific intensity assigned to each of the component reflections.

The purpose of the LaueView code, which is the basis of the work described here, is to process the raw data to obtain a set of corrected, accurate structure factor amplitudes. The code consists of three phases:

1. Prediction, in which the locations of the diffraction spots on the image and the wavelength of each reflection are determined;

2. Integration, in which the shape of the diffraction spots in each region of the image and the intensity of each spot are determined; and

3. Scaling, in which the measured intensities are corrected to account for the distortion effects discussed above and are reduced to structure factor amplitudes.

LaueView was written by one of the authors of this paper (Zhong Ren) and is described in detail by Ren and Moffat elsewhere [8, 9]. Our mission in the current project was to enhance the numerical and computational techniques used by LaueView to allow it to produce results of the same or better quality in less computing time. Faster processing of data gathered from real protein crystals will allow more efficient use of expensive experimental resources, such as synchotron beam lines, and more effective use of the efforts of the scientists involved in the experiment.

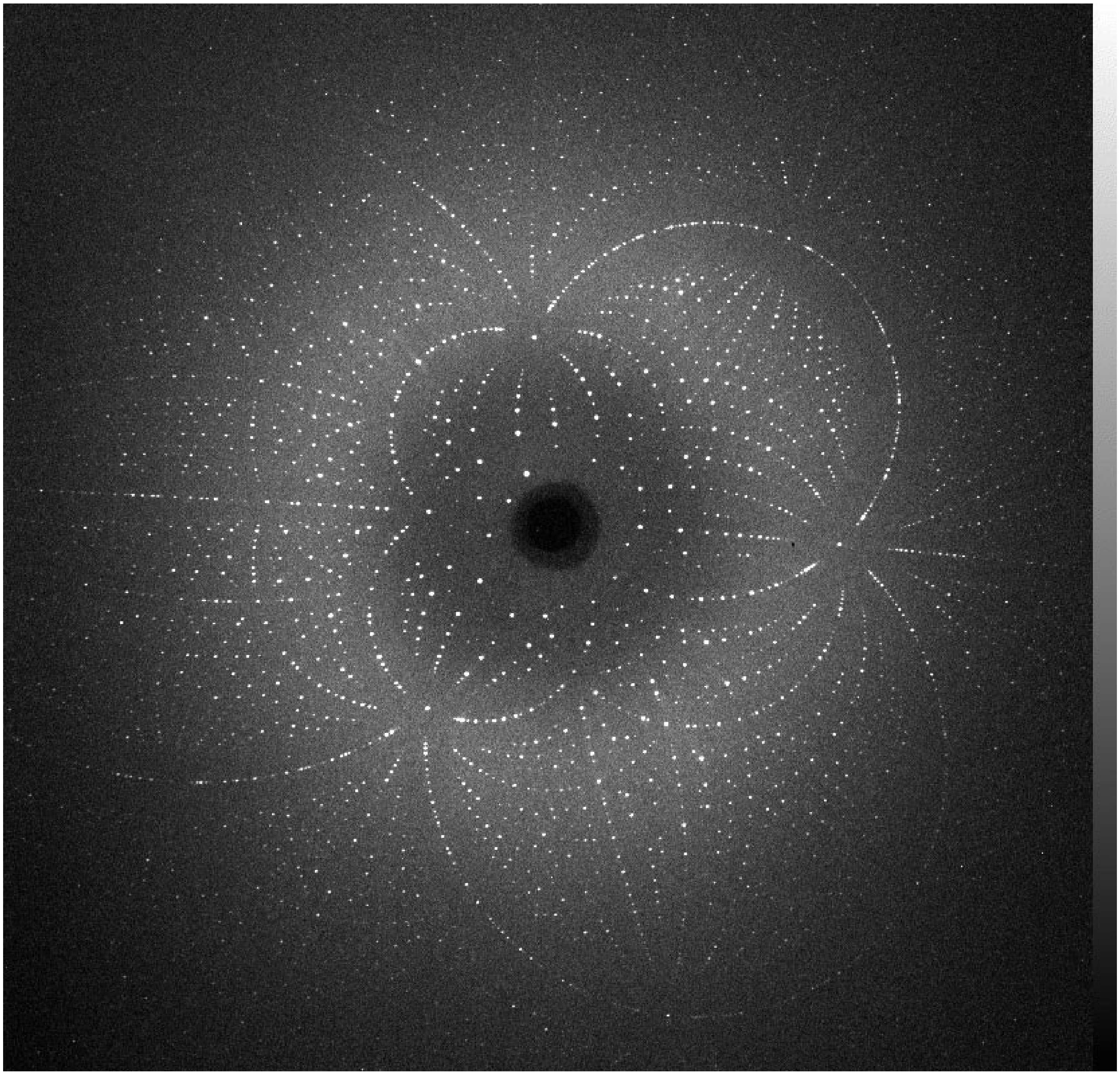

Figure 1: Laue diffraction image collected from a restrictocin crystal

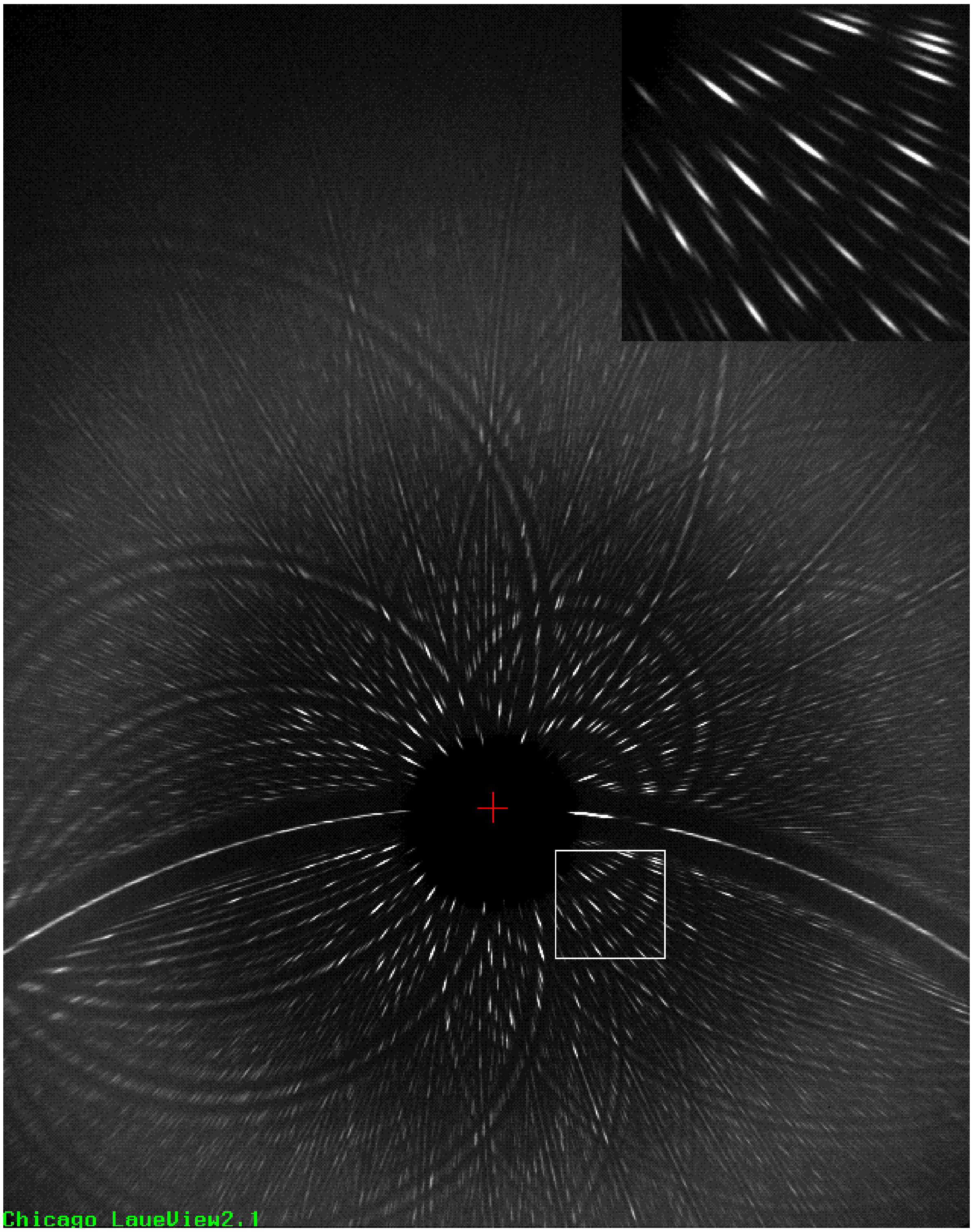

Figure 2: Laue image of thermal stable β-glucosidase crystal

The focus of our investigations was on the scaling part of the code (phase 3 above), which accounts for about half the execution time on a typical data set and involves the most challenging algorithmic issues. The integration portion of the code accounts for most of the remaining processing time, but it is naturally a parallel operation (different spots can be integrated simultaneously) and is therefore not a bottleneck on our target parallel platforms. The prediction part of the code takes relatively little time to execute.

In the Appendix, we describe the basic principles of X-ray diffraction and explain the Laue method. Although this subject is well known to physicists, we believe that our treatment below will be easy to understand for a mathematical reader who is not familiar with the topic. Section 2 describes the LaueView code, discussing both the modeling aspects and the major numerical computing issues. Section 3 describes the main computational issues in more detail and focuses on the improvements we made in these areas. Computational results on an actual data set are given in Section 4.

2 The LaueView Code

LaueView, the code with which we worked in this study, analyzes the measured intensity data from the detector and outputs the structure factor amplitudes $|F(h,k,\ell)|$for each set of Miller indices (h,k,ℓ). This section presents a brief description of LaueView. For more details, see Ren and Moffat [9].

The first phase of LaueView involves indexing and prediction of the diffraction pattern, using knowledge of the lattice and symmetry information, and of various geometric parameters of the actual experiment, such as the location of the beam center and of the crystal-to-detector distance. Good estimates of the latter parameters are available, and these estimates are refined by a least-squares data-fitting procedure based on comparison of the predicted pattern with the observed pattern. The main purpose of this phase is to determine the (h,k,ℓ) indices and the wavelength associated with each spot. Such information is essential in dealing with the spatial-overlap problem that was mentioned at the end of the appendix.

In the second phase of LaueView, the raw intensities associated with each spot are determined. Because of the spatial-overlap problem and the irregular shape of most spots, this phase requires sophisticated modeling and numerical techniques. The shapes of spots in the same area of the detector tend to be similar, though their intensities may vary widely. LaueView formulates a model for the intensity distribution in a group of overlapping spots in which a set of shared parameters (which vary only slowly across the detector space) is combined with one intensity parameter per spot, as well as a parameter that defines the average background intensity. Specifically, the intensity formula for a group of $n+1$ overlapping spots that are centered at (x_i, y_i), $i = 0, 1, \ldots, n$, with elliptical orientation φ_i, $i = 0, 1, \ldots, n$, is

$$\begin{aligned} P(x,y) &= \sum_{i=0}^{n} p_i \exp\{-E(x,y;x_i,y_i,\varphi_i;a,b,\epsilon,d_x,d_y,s_a,t_a,s_b,t_b,g_a,g_b)\} + \\ &\qquad p_x(x-x_0) + p_y(y-y_0) + p_b, \end{aligned} \tag{1}$$

where

$$\begin{aligned} &E(x,y;\hat{x},\hat{y},\hat{\varphi};a,b,\epsilon,d_x,d_y,s_a,t_a,s_b,t_b,g_a,g_b) \\ &= \left[\frac{(x-\hat{x}+d_x)\cos(\hat{\varphi}+\epsilon)+(y-\hat{y}+d_y)\sin(\hat{\varphi}+\epsilon)}{a+s_a(x-\hat{x})+t_a(y-\hat{y})}\right]^{2g_a} \end{aligned} \tag{2}$$

$$+\left[\frac{-(x-\hat{x}+d_x)\sin(\hat{\varphi}+\epsilon)+(y-\hat{y}+d_y)\cos(\hat{\varphi}+\epsilon)}{b+s_b(x-\hat{x})+t_b(y-\hat{y})}\right]^{2g_b}.$$

Note that the coordinates (x_i, y_i) and orientations φ_i, $i = 0, 1, \ldots, n$ are determined in the prediction phase. (The parameters d_x, d_y, and ϵ are corrections to x_i, y_i, and φ_i, respectively, that are uniform within the spot group and are allowed to vary only slowly across the parameter space.)

Each spot group contains 14 shared parameters—a, b, ϵ, d_x, d_y, s_a, t_a, s_b, t_b, g_a, g_b, p_x, p_y, and p_b—as well as the $n+1$ intensity parameters p_i, $i = 0, 1, \ldots, n$. To obtain values for the shared parameters, LaueView chooses a small fraction of reflections from the detector—reflections for which the signal-to-noise ratio is high, spatial overlap is at a minimum, and the spot center is well predicted. For each spot group in this sample, it performs nonlinear least-squares data fitting of the predicted intensity (1) to the observed data, in which all $n+15$ parameters are allowed to vary. The detector space is then partitioned into a number of "bins," and an averaging and smoothing procedure is applied to the shared parameters. The end result of this process is a "standard profile" for each bin that is defined by a set of 13 shared parameters that are shared by all spots within the bin. (The smoothing procedure ensures that these parameters vary only slowly across the entire detector space.) Having thus fixed the shared parameters for all reflections, we now perform linear least-squares data fitting to obtain values for the intensity parameters p_i $(i = 0, 1, \ldots, n)$ and p_b for each spot. (Note that these parameters occur only linearly in the formula (1), hence the need only for linear least squares.) Finally, the intensity I associated with a particular reflection at the point $(\hat{x}, \hat{y})$ is obtained by integrating its intensity profile $P(x, y)$ by using a standard rule for numerical integration (Simpson's rule or Gaussian quadrature), and scaling by the intensity parameter p obtained from the linear-least-squares data fit to that reflection's spot group.

In the third phase of LaueView—the scaling phase—corrections are applied to the complete set of measured intensities to account for such effects as temperature factors, radiation damage, general absorption, and, most important, the varying intensity of the incident beam across its range of wavelengths. The resulting nonlinear least-squares data-fitting problem involves a very large number of observations (of the order of 10^5–10^6 for a protein data set) and a relatively small number of parameters (typically 100–200). We recount a few details of the scaling operation here and refer the interested reader to Ren and Moffat [8, 9] for further information.

We seek a multiplicative scaling factor f to be applied to each measured intensity I, to obtain a corrected intensity fI. The factor f depends on the Miller indices (h, k, ℓ) and an index i of the particular observation of this reflection. In addition, f depends on various parameters whose values are to be recovered from the data-fitting process. It is composed of a product of 12 factors:

$$f = f_L f_P f_\lambda f_{\mathrm{isoS}} f_{\mathrm{anisoS}} f_{\mathrm{isoB}} f_{\mathrm{anisoB}} f_{\mathrm{isoD}} f_{\mathrm{anisoD}} f_A f_U f_O, \tag{3}$$

where each factor represents a correction for a different effect. These component factors include the following:

- A polarization correction f_P of the form

$$f_P = 2/(1+\cos^2(2\theta) - \sin\rho\cos 2\varphi \sin^2 2\theta),$$

  where θ and φ are the Bragg angle of reflection and the polar angle (which are functions of (h, k, ℓ)) whereas $\sin\rho$ is the X-ray-beam polarization ratio. Here, ρ is a parameter whose value is to be determined by the data-fitting process.

- An isotropic scale factor f_{isoS} with the form

$$f_{\mathrm{isoS}} = e^s,$$

where s is a parameter to be determined by data fitting.

- An anisotropic scale factor f_{anisoS}, which has one of the following two forms, depending on the desired level of sophistication:

$$f_{\text{anisoS}} = \exp(a_1 h + a_2 k + a_3 \ell), \tag{4}$$

or

$$f_{\text{anisoS}} = \exp(a_1 h + a_2 k + a_3 \ell + a_4 h^2 + a_5 k^2 + a_6 \ell^2 + a_7 hk + a_8 k\ell + a_9 \ell h), \tag{5}$$

where (h, k, ℓ) are the Miller indices. The parameters are a_i, $i = 1, 2, \ldots$.

- Isotropic and anisotropic temperature factors f_{isoB} and f_{anisoB}, respectively, defined by

$$f_{\text{isoB}} = \exp(-B \sin^2 \theta / \lambda^2),$$
$$f_{\text{anisoB}} = \exp(-B_h |\bar{\mathbf{a}}|^2 h^2 - B_k |\bar{\mathbf{b}}|^2 k^2 - B_\ell |\bar{\mathbf{c}}|^2 \ell^2),$$

where $|\bar{\mathbf{a}}|$, $|\bar{\mathbf{b}}|$, and $|\bar{\mathbf{c}}|$ are the lengths of the reciprocal lattice vectors and λ is the wavelength that produced the reflection. There are four scalar parameters: B, B_h, B_k, and B_ℓ.

The wavelength normalization factor f_λ is used to correct for the fact that the incident beam contains a spread of wavelengths, of varying intensity. The curve that relates wavelength to intensity, known as the λ *curve*, is not known directly—it must be parameterized and reconstructed by fitting our intensity data and by using our knowledge of the crystal symmetry. Redundant measurements at different wavelengths and knowledge of the symmetry are instrumental in reconstructing the λ curve. The curve is continuous, but may contain downward spikes at the absorption wavelengths of a beam-focusing mirror (see Ren and Moffat [9]).

In LaueView2.5, the λ curve is defined in terms of the Chebyshev basis functions $\cos(i \arccos z)$, $i = 1, 2, 3, \ldots$, for $z \in [-1, 1]$. We first select the range $[\lambda_{\text{min}}, \lambda_{\text{max}}]$ of measured frequencies and define a normalized frequency measure λ' by

$$\lambda' = \frac{\lambda - (\lambda_{\text{max}} + \lambda_{\text{min}})/2}{(\lambda_{\text{max}} - \lambda_{\text{min}})/2}. \tag{6}$$

The λ curve (equivalently, the correction factor f_λ) is defined as

$$f_\lambda = 10^{-10} + \exp\left[\sum_{i=1}^{n_\lambda} c_i \left(\cos(i \arccos \lambda') - \cos(i \arccos \lambda'_r)\right)\right], \tag{7}$$

where $\lambda'_r \in [-1, 1]$ is a reference frequency that fixes the scale of f_λ (through the relation $f_\lambda = 1 + 10^{-10}$ when $\lambda' = \lambda'_r$). The degree of the Chebyshev expansion (chosen by the user) is n_λ, while c_i, $i = 1, 2, \ldots, n_\lambda$, are the coefficients to be determined by fitting the data. The term 10^{-10} is introduced to keep f_λ strictly positive over the range in question.

Parameterization of the λ curve was one of the features that we changed in transitioning to LaueView3.1, as we describe in the next section.

The other factors in (3) that we do not discuss in detail are the Lorentz factor f_L, isotropic and anisotropic radiation damage factors f_{isoD} and f_{anisoD}, a general absorption correction f_A, a detector spatial-nonuniformity corrector f_U, and a detector nonlinearity corrector f_O.

For practical data sets, LaueView usually does not obtain its best results in a single least-squares minimization in which all scaling parameters are allowed to vary at once. Rather, a better minimizer is usually found by performing a sequence of runs, where only a subset of the parameters is allowed to vary in each run while the remainder are fixed. The decision about which parameters to allow to vary on each run in the sequence is left to the user; intuition from working with large data sets seems to make for good decisions.

LaueView2.5 is a Fortran 77 code of approximately 50,000 lines. Most of its arithmetic is performed in single precision. A Unix shell script is used to set up each run. The user edits this script to select the parameters to be varied and those to be fixed on a particular run, to decide the number of basis functions to be used in the parametrization of the λ curve, and to indicate whether the code should start "cold" or use the approximate solution generated by a previous run as its starting point. LaueView was developed specifically for single-processor SGI workstations. It can be used to analyze even monochromatic data if the parameters are set appropriately. A fuller description of the code (from the viewpoint of the application) can be found in the papers of Ren and Moffat [8, 9]. It was used to produce the structures reported in Ren et al. [10] and Genick et al. [2]

3 Numerical Computing Issues

LaueView uses numerous techniques from scientific computing, numerical analysis, and optimization, both in formulating its model of the diffraction process and in "solving" this model to obtain the corrected intensities. Chief among these techniques are the following.

- Curve fitting, in which the λ-curve is approximated by $e^{c(\lambda)}$, where $c(\lambda)$ is a finite linear combination of basis functions.
- Nonlinear least-squares minimization. A large problem of this type is solved during the scaling phase of LaueView to determine the parameters, other than the crystal and molecular structure, that affect the measured intensities. In addition, many small problems of this type are solved during the integration phase of LaueView to determine the profile that defines the shape, orientation, and intensity distribution within each spot.
- Numerical linear algebra, which is used in the solution of the norm-constrained linear least-squares problem that arises at each iteration of the nonlinear least-squares algorithm.

In these and other areas, we were able to enhance the performance of LaueView considerably. The previous version, LaueView2.5, evolved into a new version, LaueView3.1. In this section, we outline the major numerical operations in LaueView, highlighting the improvements we made in the new version of the code. We also describe how the code was parallelized for execution on an IBM SP multiprocessor.

3.1 Least-Squares Data Fitting

Data fitting is the process of estimating the parameters in the model of a system by trying to match a set of observations of the system as closely as possible. Suppose we denote the model by the function $\phi : \mathsf{R}^n \times \mathsf{R} \to \mathsf{R}$, where $\phi(x,t)$ is the output of the model for a given parameter vector x and data point (ordinate) $t \in \mathsf{R}$. Suppose that we have a set of m observations δ_i of the physical system taken at ordinates t_i, $i = 1, 2, \ldots, m$ (that is, at the data ordinate $t = t_i$, the observed output of the system was $\delta_i \in \mathsf{R}$). We can form a *residual vector* $r(x)$ of the differences between the model values and the observations, for a given parameter vector x, as follows:

$$r(x) = [r_i(x)]_m^{i=1}, \quad \text{where } r_i(x) = w_i[\phi(x,t_i) - \delta_i].$$

The w_i, $i = 1, 2, \ldots, m$, are a set of fixed positive weights, chosen to balance the relative importance of the different observations. They could reflect our relative confidence in each observation; those for which the measurements are known to be more accurate and less "noisy" are sometimes assigned higher weights.

In least-squares data fitting, we seek the vector x that minimizes a weighted Euclidean norm of this vector. That is, we solve the optimization problem

$$\min_x f(x) = \tfrac{1}{2} \sum_{i=1}^{m} r_i^2(x) = \tfrac{1}{2} \|r(x)\|_2^2. \tag{8}$$

Various specialized and efficient algorithms have been developed for solving this problem. Most of these methods exploit the special structure of the gradient and Hessian (the first and second derivative entities) of the function $f(x)$. The Jacobian of the vector $r(x)$—the $m \times n$ matrix of first partial derivatives—can be written as

$$J(x) = \left[\frac{\partial r_i}{\partial x_j} \right]_{i=1,\ldots,m;j=1,\ldots,n}.$$

Elementary calculus then shows that the gradient $\nabla f(x)$ and Hessian $\nabla^2 f(x)$ are as follows:

$$\nabla f(x) = \sum_{i=1}^{m} r_i(x) \nabla r_i(x) = J(x)^T r(x); \tag{9a}$$

$$\nabla^2 f(x) = J(x)^T J(x) + \sum_{i=1}^{m} r_i(x) \nabla^2 r_i(x). \tag{9b}$$

The algorithm used to solve the least-squares problems that arise both in the integration and in the scaling phases of LaueView—the Levenberg-Marquardt algorithm—can be viewed as a Newton-like method with a trust region constraint on the step length, in which the true Hessian $\nabla^2 f$ is approximated by the first term $J^T J$ in its definition above (here and subsequently, for conciseness, we omit the argument x). In many situations, good steps can be obtained by making this approximation. The second term in (9b) may be dominated by the first term $J^T J$ when the residuals r_i, $i = 1, 2, \ldots, m$, are small at the solution, or when the residual functions r_i are nearly linear, or when cancellation occurs in the summation. Even when the second term is not insignificant, the approximation $J^T J$ has the advantages of being positive semidefinite and of having similar scaling to the true Hessian.

Our implementation of the Levenberg-Marquardt algorithm in LaueView3.1 follows that of Moré [5]. From our current point x, we obtain a candidate step $\delta \in \mathbb{R}^n$ by solving the following subproblem:

$$\min_{\delta \in \mathbb{R}^n} \tfrac{1}{2} \|r + J\delta\|^2 \quad \text{subject to} \quad \|D\delta\|_2 \leq \Delta, \tag{10}$$

where the scalar Δ is known as the trust-region radius, and D is a diagonal scaling matrix $D = \operatorname{diag}(d_1, d_2, \ldots, d_n)$ with positive diagonal elements. Ideally, D should be chosen so that the sensitivity of the function f to perturbations in the values of each scaled variable $(Dx)_i$ is roughly the same. (In LaueView3.1, we set $D_{ii} = \max((J^T J)_{ii}, 1)$.) It is well known (see, for example, Moré and Sorensen [6]) that the solution of (10) satisfies the equation

$$[J^T J + \gamma D^2]\delta = -J^T r, \tag{11}$$

or, equivalently,

$$[D^{-1}J^TJD^{-1}+\gamma I](D\delta) = -D^{-1}J^Tr, \tag{12}$$

for some value $\gamma > 0$. We compute γ and the corresponding δ by the following procedure. First, we compute the matrix $D^{-1}J^TJD^{-1}$ explicitly and compute the factorization

$$D^{-1}J^TJD^{-1} = QSQ^T, \tag{13}$$

where $Q = [q_1 \,|\, q_2 \,|\, \ldots \,|\, q_n]$ is orthogonal and $S = \text{diag}(s_1, s_2, \ldots, s_n)$, with

$$s_1 \geq s_2 \geq \cdots \geq s_n \geq 0.$$

Since

$$(D^{-1}J^TJD^{-1}+\gamma I) = Q(S+\gamma I)Q^T,$$

we can write the solution of (12) explicitly as

$$D\delta = -\sum_{i=1}^{n} \frac{q_i^T D^{-1}J^T r}{s_i+\gamma} q_i. \tag{14}$$

By orthonormality of the vectors $q_1, q_2, \ldots, q_n$, we have that

$$\|D\delta\|^2 = \sum_{i=1}^{n} \frac{(q_i^T D^{-1}J^T r)^2}{(s_i+\gamma)^2}. \tag{15}$$

If the value $\gamma = 0$ yields a scaled step norm $\|D\delta\|$ smaller than Δ, then we set $\gamma = 0$ in (11) and compute δ from the formula (14) with $\gamma = 0$. Otherwise, the problem of finding the appropriate $\gamma(\Delta)$ such that $\|D\delta\| = \Delta$ is equivalent to finding a root of the scalar function

$$R_\Delta(\gamma) = \sum_{i=1}^{n} \frac{(q_i^T D^{-1}J^T r)^2}{(s_i+\gamma)^2} - \Delta^2,$$

which is usually a monotonically decreasing function of γ. A specialized root finding algorithm can be applied to $R_\Delta(\gamma)$ to find an approximately optimal γ (see Moré [5] for details). Then, recovering the step δ from (14), it computes the ratio ρ of actual function decrease along this step to the decrease predicted by the model in (10), that is,

$$\rho = \frac{f(x) - f(x+\delta)}{f(x) - \frac{1}{2}\|r(x) + J(x)\delta\|^2}. \tag{16}$$

If this ratio is larger than a small positive number ϵ ($\epsilon = 10^{-3}$, say), the step is accepted, and we set $x^+ = x + \delta$. If ρ is close to its ideal value of 1 ($\rho > .75$, say), and if the trust-region constraint is binding (that is, $\|D\delta\| = \Delta$), we enlarge the trust region for the next iteration by setting $\Delta \leftarrow 2\Delta$. If, on the other hand, ρ falls below the acceptance threshold ϵ, we reject the step δ decrease Δ (by a factor of 4, say), and re-solve (10) to obtain a new δ.

In the scaling part of LaueView, a typical value of m for a large data set is 10^5, while n is typically of the order of 10^2. The Jacobian J is large, "long and skinny," and not particularly sparse. Some implementations of Levenberg-Marquardt (for example, the one described by Moré [5]) store J explicitly, compute a QR factorization, and use the R factor in the subsequent calculations needed to identify $\gamma(\Delta)$. In LaueView, J is too large to store explicitly. Instead, LaueView calculates the residual vector r and the Jacobian J row by row (that is, it calculates the quantities r_i and ∇r_i in sequence for $i = 1, 2, \ldots, m$) and accumulates the products $J^T J$ and $J^T r(x)$ by using the formulae

$$J^T J = \sum_{i=1}^{m} \nabla r_i (\nabla r_i)^T, \qquad J^T r = \sum_{i=1}^{m} r_i \nabla r_i. \tag{17}$$

Because the evaluations of r_i and ∇r_i are independent for different values of i, they can be carried out in parallel. We discuss this point further in Section 3.4.

A variant of the Levenberg-Marquardt algorithm obtained from the book *Numerical Recipes* [7] was used in the earlier version of the code, LaueView2.5. This variant does not make use of a trust-region strategy but rather manipulates the parameter γ explicitly. This parameter is increased when the candidate step δ is unsatisfactory, and decreased when a successful step is taken. The alternative trust-region implementation that we described above allows more direct control over the length of the computed steps. A significant disadvantage of the algorithm described in [7] is that it always evaluates the first derivatives ∇r_i whenever the residual values r_i are evaluated. If the candidate iterate is subsequently rejected (when it does not lead to a significant decrease in the function value over the current iterate), this derivative evaluation is wasted. If, as in LaueView and in many other applications, the derivative is relatively expensive to calculate, the overall wastage in computational effort may be considerable. In the new version, LaueView3.1, we separated evaluation of the residual r from evaluation of the Jacobian J, and evaluated the latter only after a point was accepted as the new iterate.

3.2 Linear Algebra Issues

The most important basic linear algebra calculations in LaueView are accumulation of the sums in (17) to form $J^T J$, calculation of the factorization (13), and repeated solution of the system (14) for δ. In LaueView2.5, the terms in (17) were calculated and accumulated in single-precision arithmetic. This operation appeared to cause difficulties in the Levenverg-Marquardt implementation, because the code usually terminated by increasing γ to a very large number while still failing to achieve descent in the objective function, suggesting that the computed gradient $-J^T r$ was not a descent direction for the objective function f. In LaueView3.1, we modified the code so that each residual r_i and its derivatives were still evaluated in single precision, but the results were transferred to double-precision variables, and the quantities $J^T J$ and $J^T r$ were accumulated and stored in double precision. On modern computer architectures, double-precision arithmetic is not much (if any) more expensive than single precision, and the additional storage needed for double-precision variables in our scheme is not significant.

LaueView2.5 performed a factorization of the matrix $J^T J + \gamma D^2$ by using either singular value decomposition or Gaussian elimination, and used the resulting factors to solve (11) for the candidate step δ. LaueView3.1 finds the decomposition (13) (which is equivalent to the svd for this symmetric matrix) by using the LAPACK routine DSYEV and obtains the candidate steps from (14). This revised strategy is slightly more economical because it exploits symmetry in the calculation of (13) and avoids recalculation of the factorization for each different value of γ. This part of the computation is considerably less expensive than calculation of the residuals r_i and their derivatives.

3.3 Approximating the λ Curve with Local Basis Functions

In LaueView2.5, the Chebyshev basis functions $\cos(i \arccos z)$, $i = 1, 2, 3, \ldots$, were evaluated as written for many values of z, by using Fortran library functions to evaluate the cos and arccos functions. In a later version (LaueView3.0, which predates LaueView3.1), we replaced this technique by a more efficient, well known recurrence relation. Given $z \in [-1, 1]$, we define

$$c_1 = \cos(\arccos z) = z, \quad s_1 = \sin(\arccos z) = \sqrt{1 - z^2},$$

and recur by using the formulae

$$c_{i+1} \leftarrow c_i c_1 - s_i s_1, \quad s_{i+1} \leftarrow \sqrt{1 - c_{i+1}^2}, \quad i = 1, 2, \ldots.$$

Global basis functions such as Chebyshev are not particularly well suited to the representation of the λ curve, since (as noted above) this curve is not particularly smooth and contains downward spikes at the absorption wavelengths of the focusing mirror. Moreover, such functions give rise to a Jacobian matrix that is almost fully dense and quite expensive to evaluate. In LaueView3.1, we replaced them by piecewise-quadratic basis functions with local support defined on a uniform mesh over the user-defined range of wavelengths spanned by the data. We then replace the definition (7) of f_λ by

$$f_\lambda = 10^{-10} + \exp\left[\sum_{i=1}^{n_\lambda} c_i \left(p_i(\lambda') - p_i(\lambda'_r)\right)\right], \tag{18}$$

where $\lambda' \in [-1, 1]$ is the normalized frequency from (6), λ'_r is the reference frequency, and

$$p_i(\lambda') = \begin{cases} (\lambda' - a_{i-3})^2/(6h^3) & \lambda' \in [a_{i-3}, a_{i-2}], \\ 1/(4h) - (\lambda' - a_{i-3/2})^2/(3h^3) & \lambda' \in [a_{i-2}, a_{i-1}], \\ (\lambda' - a_i)/(6h^3) & \lambda' \in [a_{i-1}, a_i], \end{cases}$$

where

$$h = 2/(n_\lambda - 2), \quad a_i = -1 + ih, \quad i = -3, -2, \cdots, n_\lambda.$$

3.4 Parallel Implementation

The most expensive part of the calculation in the scaling phase of LaueView is evaluation of the quantities $r^T r$, $J^T J$ and $J^T r$, at a given parameter vector x. LaueView evaluates these terms in the manner suggested by the formulae (17). That is, for each value of the index i in turn, it evaluates r_i^2, $r_i \nabla r_i$, and $\nabla r_i (\nabla r_i)^T$ and accumulates these quantities in the appropriate data structures.

A parallel version of this process proceeds in the obvious way: The indices $i = 1, 2, \ldots, m$ are "dealt out" to the P available processors, so that each processor receives an approximately equal number of indices. Processor j forms its own partial sums $J_{[j]}^T J_{[j]}$ and $J_{[j]}^T r_{[j]}$, where $[j]$ denotes the subset of indices received by processor j. A global summation operation is performed to obtain $J^T J = \sum_{j=1}^P J_{[j]}^T J_{[j]}$. Parallel evaluation $r^T r$ and $J^T r$ takes place similarly. The other operations associated with LaueView, including computation of the candidate Levenberg-Marquardt step, take place concurrently (and redundantly) on all processors; their relatively low computational cost makes it not worthwhile to parallelize them.

4 Computational Results

In this section we report on the effects of our improvements to LaueView, as measured by its performance on a real data set from a crystal of photoactive yellow protein, for which structure results obtained with LaueView2.5 are presented in [2]. The value of m for this set is approximately $120,000$,

Table 1: Parameter Settings for PYP data set

Parameter	1	2	3	4	4S	6
Hot start? (starting set)	N	Y (1)	Y (1)	Y (1)	N	N
Image weighting	N	Y	Y	Y	Y	N
Lorentz f_L	Y	Y	Y	Y	Y	Y
Polarization f_P	N	Y	Y	Y	Y	N
Wavelength f_λ	Y	N	N	Y	Y	Y
Isotropic scaling f_{isoS}	N	Y	N	Y	Y	Y
Anisotropic scaling f_{anisoS} (4)	N	N	N	N	N	N
Anisotropic scaling f_{anisoS} (5)	N	N	N	N	N	N
Isotropic scaling f_{isoB}	N	N	Y	Y	Y	Y
Anisotropic scaling f_{anisoB}	N	N	N	N	N	N

while the number of parameters n is small, between about four and seventy in our experiments. We focus on the improvements in computational performance in this section, rather than the scientific findings.

Table 1 indicates the parameter settings for the six runs that we performed on the PYP data set. Most of the rows correspond to parameters while the columns indicate the runs. The entry Y or N indicates whether the parameter in question was allowed to vary (Y) or held fixed (N) during the run in question. The first row indicates whether or not a hot start was performed for the run in question, using as a starting point the output of another run. Runs 1, 4S, and 6 do not use a hot start, while runs 2, 3, and 4 use the final point attained by run 1 as their starting point. For instance, run 3 of LaueView2.5 uses the output of run 1 of LaueView2.5 as its starting point, while run4 of LaueView3.1 uses the output of run 1 of LaueView3.1 as its starting point.

We performed experiments on two computational platforms. The first was an SGI Onyx2 Reality Monster running IRIX 6.4, equipped with sixteen MIPS R10000 processors (of which we used just one) and 4 GB of memory. The second was an 80-node IBM SP in which each node is an RS/6000 workstation equipped with a 120 MHz P2SC chip and 256 MB of memory. On the SGI, the code used SGI's XFS file system to do I/O from disk. On the IBM, the parallel input/output file system was used. The use of these advanced file systems reduced the time to completion considerably (without affecting the CPU time) because the code needs to write (and in some cases to read) very large files. On the IBM SP, about 100 seconds of CPU time is needed to write an 85 MB output file that contains the corrected intensity data, independently of the number of processors used and of the particular version of LaueView. In the cases of runs 2, 3, and 4, a file of this size is also read at the start of the computation to provide "hot start" data.

4.1 Numerical Improvements

Table 2 summarizes the computational performance of the three different versions of LaueView on the SGI machine. LaueView2.5 is the original version of the code prior to numerical improvements, LaueView3.1 incorporates all the improvements described above, while LaueView3.0 is identical to LaueView3.1 except that it continues to use the Chebyshev basis functions of LaueView2.5 instead of B-spline basis functions. We tried two choices $n_\lambda = 32$ and $n_\lambda = 64$ for the number of local basis functions in LaueView3.1. In LaueView2.5 and 3.0, 64 Chebyshev basis functions were used in all runs.

Table 2: CPU Times (seconds) for Different Versions of LaueView on SGI Reality Monster

Data Set	LaueView2.5		LaueView3.0		LaueView3.1 ($n_\lambda = 64$)		LaueView3.1 ($n_\lambda = 32$)	
	time (s)	optimal f	time (s)	optimal f	time (s)	optimal f	time (s)	optimal f
1	2096	1045598	411	1042412	257	1006824	191	1017473
2	3725	485746	260	491415	376	478604	247	509350
3	7700	576934	529	575237	343	562133	386	591343
4	failed		1618	466239	698	462022	559	492059
4S	not tested		4034	467767	2380	461494	1617	491801
6	10841	633712	2530	639705	1525	598617	1171	610565

Table 3: Parallel Performance of LaueView3.1 on IBM-SP Multiprocessor

Data Set	Processors	Time (s)	Speedup	Approx. Speedup (excluding data output)
1	1	551		
1	2	377	1.5	1.6
1	4	274	2.0	2.6
6	1	4443		
6	2	2522	1.8	1.8
6	4	1334	3.3	3.5
6	8	765	5.8	6.5
6	16	485	9.2	11.3
6	32	351	12.7	17.3

From Table 2, we see that LaueView3.0 obtains similar final objective function values to LaueView2.5 in considerably less CPU time. On data sets for which both codes produced a result, the improvement in CPU time is a factor of between about 5 and 14. LaueView3.0 is also considerably more robust, as demonstrated by its convergence on runs 4 and 4S (yielding the smallest objective function values in the first two columns) where LaueView2.5 failed.

The use of local basis functions (LaueView3.1) results in a further improvement. For LaueView3.1, smaller final objective values were found in all cases, and CPU requirements decreased significantly in most cases. Further decreases in CPU time can be noted when n_λ is decreased to 32, at the cost of slight increases in the optimal objective value.

Another lesson we draw from this table is the usefulness of the strategy of finding a good minimizer by performing a sequence of runs in each of which just a subset of parameters is allowed to vary. Runs 4 and 4S allow the same parameters to vary, the only difference being that run 4 uses as its starting point the result of run 1, while run 4S starts cold. The combined strategy of run 1 followed by run 4 finds a slightly smaller objective value in a total run time of half that of run 4S ($411 + 1618 = 2029$ seconds vs 4034 seconds).

4.2 Parallelization

Results of the parallel code running on the IBM SP are presented in Table 3. We show results on runs 1 and 6—a short run and a longer one. LaueView3.1 was used with the number of basis functions set to 64. In all cases, the results of the computation are independent of the number of processors used.

Note that the single-processor times are longer than on the SGI platform by a factor of two to three, since the SGI nodes are more powerful. Since, as mentioned above, we parallelized only the critical section of the code in which the matrix $J^T J$ and the vector $J^T r$ are evaluated and accumulated, the speedups are considerably less than linear in the number of processors; the non-parallel parts of the computation (particularly the 100 seconds spent in writing the output file to disk) become relatively more significant as the number of processors is increased. However, the wall-clock time is reduced considerably on multiple processors. The final column in Table 3 indicates the speedup figure obtained when the 100 seconds spent on writing the output file is subtracted from the total CPU time for each run. A more sophisticated parallel code could parallelize this operation by having each processor write a section of the output to its own file, while any subsequent run could read these files in parallel, in an analogous fashion. We feel that the run-time advantages obtained with our current parallelization technique are sufficient to meet the needs of experimentalists for faster turnaround time, however.

Acknowledgments

We thank Joe Czyzyk and Madhu Nayakkankuppam for their help in the earlier stages of this project, both in investigating the scientific issues and in working with the LaueView code. We also thank Keith Moffat for valuable comments on an earlier draft.

A Outline of X-ray Diffraction

Diffraction of X-rays by a crystal admits a beautiful mathematical explanation in terms of lattice theory, simple geometry, and other classical tools. Here we briefly summarize of the background theory for our study.

A *lattice* is an infinite, regular array of points in a space of some given dimension that satisfies the property that its geometry relative to any point in the array is independent of the particular choice of point. A crystal is a collection of molecules arranged in a (finitely truncated) three-dimensional lattice. That is, if we choose some point in the molecule in question as a reference point, then the array of these points in space would form a three-dimensional lattice if extended infinitely in all directions.

A three-dimensional lattice can be characterized by a set of three basis vectors $\mathbf{a}$, $\mathbf{b}$, and $\mathbf{c}$ in R^3, where each point in the lattice can be expressed as

$$x\mathbf{a} + y\mathbf{b} + z\mathbf{c}, \quad \text{where } x,\ y, \text{ and } z \text{ are integers.} \tag{19}$$

We can assume that $\mathbf{a}$, $\mathbf{b}$, and $\mathbf{c}$ are linearly independent; otherwise, the lattice collapses to one of lower dimensionality. The parallelepiped whose sides are the vectors $\mathbf{a}$, $\mathbf{b}$, and $\mathbf{c}$ is referred to as the *unit cell.* The 3×3 matrix A assembled from these basis vectors, namely,

$$A = [\mathbf{a} : \mathbf{b} : \mathbf{c}],$$

is nonsingular by our linear independence assumption. Another useful concept is that of the *reciprocal lattice*, which is a lattice of the same dimension characterized by three basis vectors $\bar{\mathbf{a}}$, $\bar{\mathbf{b}}$, and $\bar{\mathbf{c}}$ with the following properties:

$$\begin{array}{lll} \mathbf{a} \cdot \bar{\mathbf{a}} = 1, & \mathbf{a} \cdot \bar{\mathbf{b}} = 0, & \mathbf{a} \cdot \bar{\mathbf{c}} = 0, \\ \mathbf{b} \cdot \bar{\mathbf{a}} = 0, & \mathbf{b} \cdot \bar{\mathbf{b}} = 1, & \mathbf{b} \cdot \bar{\mathbf{c}} = 0, \\ \mathbf{c} \cdot \bar{\mathbf{a}} = 0, & \mathbf{c} \cdot \bar{\mathbf{b}} = 0, & \mathbf{c} \cdot \bar{\mathbf{c}} = 1, \end{array} \tag{20}$$

where "$\cdot$" denotes the standard (Euclidean) inner product. It is easy to see that $\bar{\mathbf{a}}$, $\bar{\mathbf{b}}$, and $\bar{\mathbf{c}}$ are simply the columns of the matrix A^{-T}, that is,

$$[\bar{\mathbf{a}} : \bar{\mathbf{b}} : \bar{\mathbf{c}}] = A^{-T}.$$

In Figure 3, we illustrate a lattice together with its basis vectors. This figure, as well as all our other illustrations, uses a two-dimensional geometry for simplicity. The extension to three

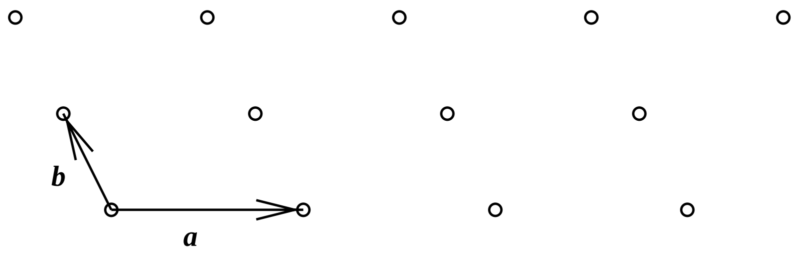

Figure 3: Two-dimensional lattice generated by basis vectors **a** and **b**

dimensions is in all cases easy to envisage. Note in particular that the difference vector **R** between any two lattice points also has the form (19), that is, it is expressible as

$$\mathbf{R} = x\mathbf{a} + y\mathbf{b} + z\mathbf{c}, \quad \text{where } x,\ y, \text{ and } z \text{ are integers.} \tag{21}$$

Each molecule in the crystal contains a number of electrons, arranged in a cloud about the atomic nuclei. When each of these electrons encounters the incident X-ray, the electron is set in motion and becomes an oscillating dipole—and therefore a source of secondary radiation. We refer to this process as "scattering" of the X-rays. Interference between the X-rays scattered from the electrons in the crystal gives rise to the diffraction patterns observed on the detector. Some scattering may also take place from the nuclei, but its amplitude is usually much smaller and can be neglected for our purposes.

To describe why certain scattering directions are directions of constructive interference, we make the (temporary) simplifying assumption that each unit cell contains a single scattering center, which is in the same location in each cell. These scattering centers themselves make up a lattice that is characterized by the same basis vectors **a**, **b**, and **c**. We now outline a geometric argument to identify the scattering directions and X-ray wavelengths for which the beams scattered from all these centers are in phase, and so yield spots on the detector.

Consider *any two* scattering centers, as shown in Figure 4. Because both are points in the lattice, the vector displacement **R** between them will have the form (21). Suppose that the incident beam has direction **t**, which we express in terms of the reciprocal lattice basis by

$$\mathbf{t} = x_t\bar{\mathbf{a}} + y_t\bar{\mathbf{b}} + z_t\bar{\mathbf{c}}, \qquad \|\mathbf{t}\| = 1/\lambda, \tag{22}$$

for some coefficients x_t, y_t, and z_t. (The normalization condition $\|t\| = 1/\lambda$ ensures that each direction is uniquely specified by the coefficient triple (x_t, y_t, z_t).) Suppose we investigate a particular direction of scattering **s**, also defined in terms of the reciprocal basis vectors with the same normalization condition as in (22) by

$$\mathbf{s} = x_s\bar{\mathbf{a}} + y_s\bar{\mathbf{b}} + z_s\bar{\mathbf{c}}, \qquad \|\mathbf{s}\| = 1/\lambda. \tag{23}$$

In Figure 5, we illustrate scattering in the direction **s** from the two lattice points separated by the displacement **R** of the form (21). The important point to note is that the path lengths traversed by the two beams differ slightly. The scattered beams will remain in phase provided that *the difference in path length is an integer multiple of the wavelength* λ. From the diagram, we see that the two different portions of the paths can be measured by dropping perpendiculars from one path to the other, and simple geometry indicates that these lengths are $\lambda\mathbf{R}\cdot\mathbf{t}$ and $\lambda\mathbf{r}\cdot\mathbf{s}$, respectively. We can therefore express the difference in path length analytically by

$$\lambda\mathbf{R}\cdot(\mathbf{s}-\mathbf{t}). \tag{24}$$

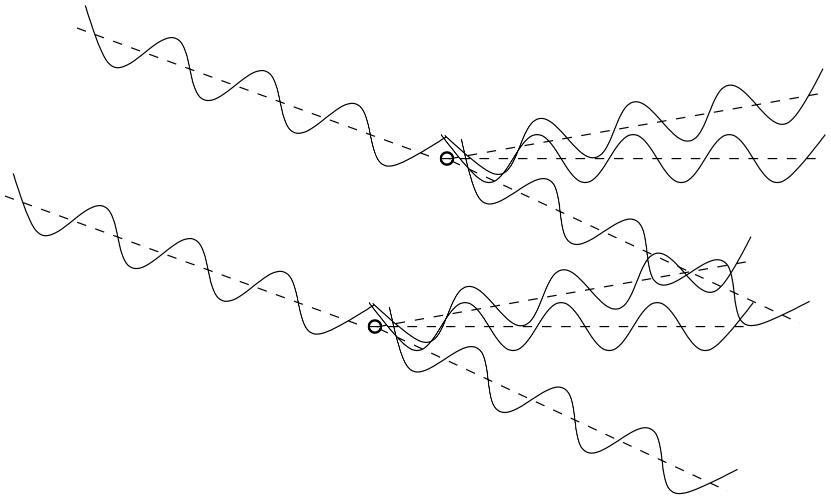

Figure 4: Diffraction from two points in the lattice (beams diffract in all directions from all points; we show just a few directions here)

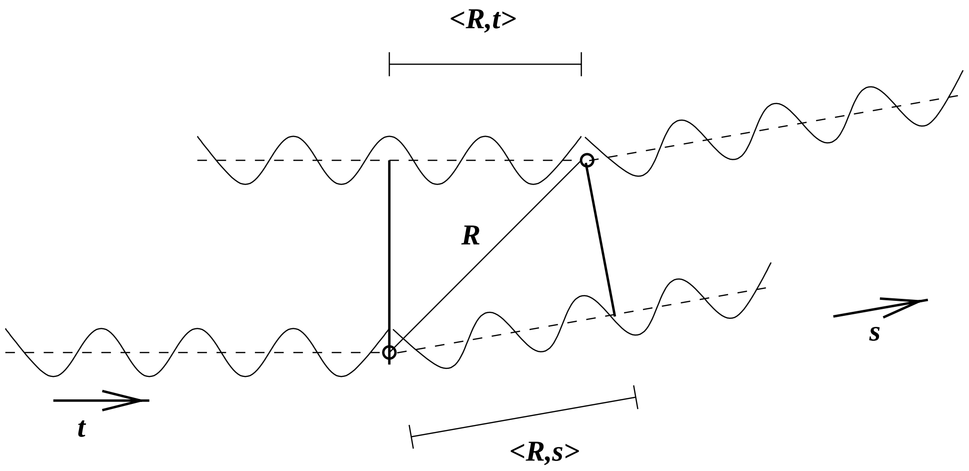

Figure 5: Beam from direction **t** diffracted in in direction s from two lattice points separated by **R**, showing difference in path length

Our requirement that this difference is an integral multiple of the wavelength can be expressed as

$$\begin{aligned}\lambda \mathbf{R}\cdot(\mathbf{s}-\mathbf{t}) &= \lambda(x\mathbf{a}+y\mathbf{b}+z\mathbf{c})\cdot[(x_s-x_t)\bar{\mathbf{a}}+(y_s-y_t)\bar{\mathbf{b}}+(z_s-z_t)\bar{\mathbf{c}}] \\ &= \lambda x(x_s-x_t)+\lambda y(y_s-y_t)+\lambda z(z_s-z_t) \\ &= \lambda m, \qquad \text{for some integer } m, \qquad (25)\end{aligned}$$

where we used the relations (20) to derive the second equality. Recall that if **s** is to yield a bright spot on the detector, the relation (25) must hold for *all* pairs of points in the lattice, that is for *all integers* x, y, and z. This is possible only if the coefficients of **s** satisfy the relations

$$x_s - x_t = h, \quad y_s - y_t = k, \quad z_s - z_t = \ell, \quad \text{for } h,\ k, \text{ and } \ell \textit{ integers}. \qquad (26)$$

The integer triple (h, k, ℓ), along with the fixed direction **t** of the incident beam, completely characterizes the direction **s**. We refer to (h, k, ℓ) as the *Miller indices.*

To summarize, we conclude from (23) and (26) that the direction **s** will produce a spot on the detector if satisfies the following conditions for some set of Miller indices (h, k, ℓ):

$$\mathbf{s} = (x_t+h)\bar{\mathbf{a}}+(y_t+k)\bar{\mathbf{b}}+(z_t+\ell)\bar{\mathbf{c}}, \qquad (27a)$$

$$||(x_t+h)\bar{\mathbf{a}}+(y_t+k)\bar{\mathbf{b}}+(z_t+\ell)\bar{\mathbf{c}}|| = 1/\lambda. \qquad (27b)$$

These relations constitute *Bragg's law.*

We can illustrate these conditions via a device known as an *Ewald sphere*, a two-dimensional version of which is plotted in Figure 6. The sphere has radius $1/\lambda$, so from the normalization conditions in (22) and (23), both **s** and **t** must lie on its surface. The grid of points represents the reciprocal lattice centered at the vector **t**. Geometrically stated, then, condition (27) says that the vector **s** is a direction of constructive interference only if it both corresponds to one of the lattice points in Figure 6 *and* lies on the surface of the Ewald sphere. The figure suggests that just a few directions satisfy this condition. It also clarifies why radiation whose wavelength is of the same order as the basis vectors **a**, **b**, and **c** is required to produce diffraction. If a longer wavelength λ is used, the Ewald sphere in Figure 6 shrinks. When the sphere becomes smaller than the spacing between the lattice points, it will not intersect the lattice, thereby producing no directions of constructive interference.

Laue diffraction images, however, contain many more spots than the description above would suggest. There appear to be many directions s, each characterized by the Miller indices (h, k, ℓ) in (27), for which a spots appears on the detector. The reason is that the incident beam consists not just of a single wavelength λ, as assumed in the description above, but a whole range of wavelengths. Typically, λ takes on a range of values between 0.3Å and 2.0Å, depending on the synchrotron source and its insertion device. Because of this property, we have not just one Ewald sphere as in Figure 6, but a continuum of spheres of varying radii $1/\lambda$ and changing origin, all of which have t on their boundary and make the same tangent with this vector. The situation is depicted in Figure 7, where the shaded area indicates the space traversed by the continuum of Ewald spheres. Each reciprocal lattice point lying in this shaded region lies on the boundary of an Ewald sphere for some $\lambda \in [\lambda_{\min}, \lambda_{\max}]$, and so gives rise to a spot on the detector. Comparison of Figures 6 and 7 suggests that we can expect many more spots in Laue diffraction than in diffraction with a monochromatic beam.

Having described why spots appear on the detector, we now outline the reasons for the differences in brightness between the spots—differences that allow a density map of the electron cloud to be constructed. Much of this effect can be attributed to the fact that scattering does *not* take place from isolated scattering centers arranged in a lattice (as we assumed for simplicity above) but rather

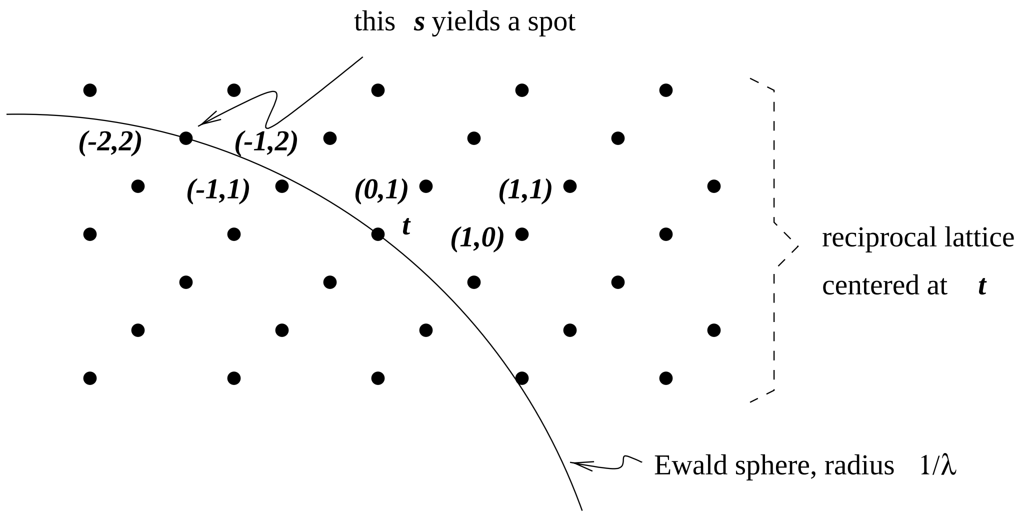

Figure 6: The Ewald sphere, in two dimensions. Direction $\mathbf{s}$ yields a spot only if it lies on the sphere of radius $1/\lambda$ *and* is a point on the reciprocal lattice originating at $\mathbf{t}$. Labels on some points show their Miller indices, which contain just two components in this two-dimensional example.

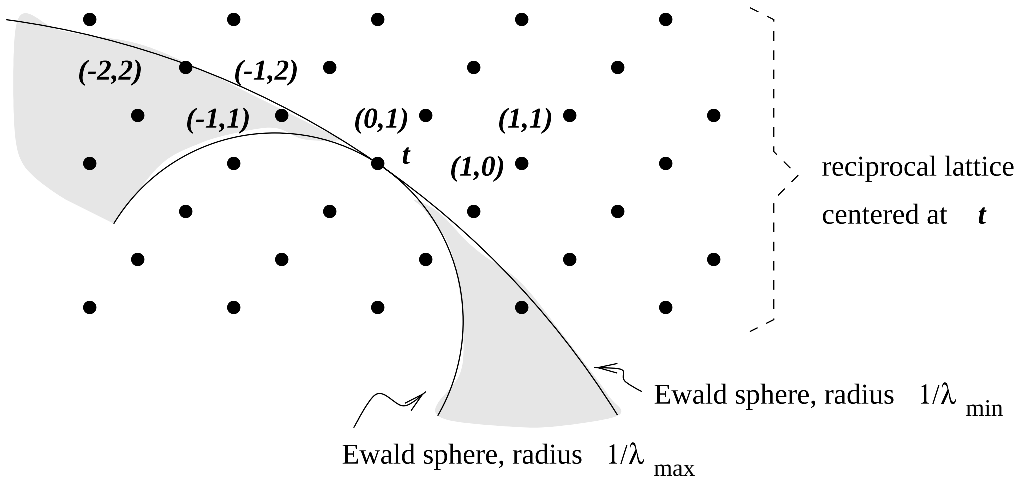
(-2,2)
(-1,2)
(-1,1)
(0,1)
(1,1)
t
(1,0)
reciprocal lattice centered at *t*
Ewald sphere, radius 1/λ min
Ewald sphere, radius 1/λ max

Figure 7: In Laue diffraction, incident radiation has wavelength in the range $[\lambda_{\min}, \lambda_{\max}]$, yielding a continuum of Ewald spheres with radii between $1/\lambda_{\max}$ and $1/\lambda_{\min}$. Any reciprocal lattice points lying between these spheres (shaded area) yields a spot on the detector.

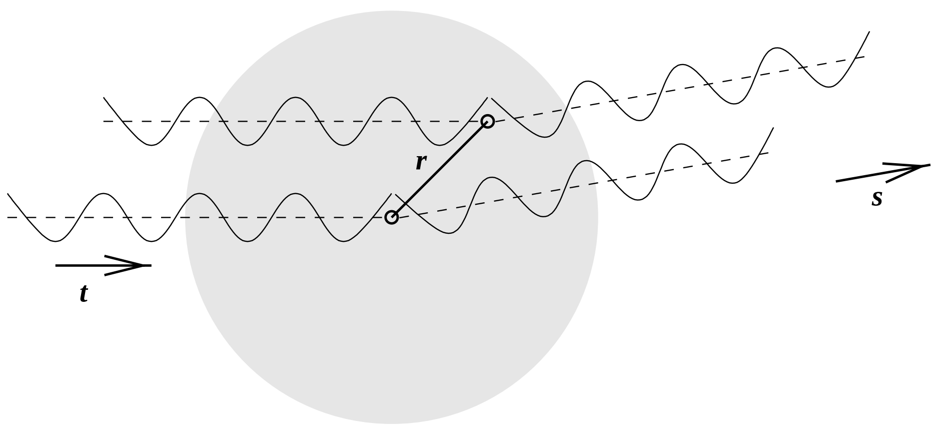

Figure 8: Scattering in the same direction **s** from the central reference point in the electron cloud and another point at a displacement **r**

from any location in the electron cloud. As a result, the scattered rays will generally not be perfectly in-phase even along the directions **s** that satisfy (27), and the intensity of the resulting spot indicates the deviation from the idealized situation of point diffractors described earlier.

To explain this effect, we first consider the case in which the point diffractor is replaced by a spherically symmetric electron cloud associated with a single atom. We use $Q(|\mathbf{r}|)$ to denote the electron density function for this cloud, where $\mathbf{r}$ indicates the displacement from the atomic nucleus (the center of the cloud), so that $|\mathbf{r}|$ denotes the radial distance. Figure 8 shows scattering of incident rays with the same incidence and scattering directions $\mathbf{t}$ and $\mathbf{s}$ from two points in the cloud. One is scattered from the central reference point, and the other from another point at a displacement $\mathbf{r}$. By using the same argument as the one that led to (24), we can show that the difference in path length between the two rays is

$$\lambda \mathbf{r} \cdot (\mathbf{s} - \mathbf{t}),$$

leading to a phase difference of

$$\left(\frac{2\pi}{\lambda}\right) \lambda \mathbf{r} \cdot (\mathbf{s} - \mathbf{t}) = 2\pi \mathbf{r} \cdot (\mathbf{s} - \mathbf{t}). \tag{28}$$

By integrating over all displacement vectors $\mathbf{r}$ in the cloud and scaling by the density, we deduce that the amplitude of the radiation scattered by the whole cloud is

$$f \stackrel{\text{def}}{=} \int_{\mathcal{S}} Q(|\mathbf{r}|) \exp(2\pi i \mathbf{r} \cdot (\mathbf{s} - \mathbf{t})) d\mathbf{r}, \tag{29}$$

where $\mathcal{S}$ denotes the sphere of integration. The quantity f is known as the *atomic scattering factor* for the atom in question. Because of spherical symmetry, we can assume without loss of generality that

$$\mathbf{t} = (1/\lambda)(1, 0, 0)^T, \quad \mathbf{s} = (1/\lambda)(\cos 2\theta, \sin 2\theta, 0)^T,$$

where 2θ is the angle of scattering. (The factor 2 is introduced for convenience.) Elementary trigonometric relationships then imply that

$$\mathbf{s} - \mathbf{t} = 2(\sin\theta/\lambda)(-\sin\theta, \cos\theta, 0)^T. \tag{30}$$

We now perform an orthogonal change of variables to $\hat{\mathbf{r}}$, where

$$\hat{\mathbf{r}} = \Pi \mathbf{r}, \quad \text{where} \quad \Pi = \begin{bmatrix} -\sin\theta & \cos\theta & 0 \\ \cos\theta & \sin\theta & 0 \\ 0 & 0 & -1 \end{bmatrix},$$

and note that $|\hat{\mathbf{r}}| = |\mathbf{r}|$ and that the domain of integration $\mathcal{S}$ is unchanged. Using (30), we can now rewrite (29) as

$$\begin{aligned} f &= \int_{\mathcal{S}} Q(|\hat{\mathbf{r}}|) \exp(2\pi i(\Pi^T \hat{\mathbf{r}}) \cdot (\mathbf{s} - \mathbf{t})) d\hat{\mathbf{r}} \\ &= \int_{\mathcal{S}} Q(|\hat{\mathbf{r}}|) \exp\left(4\pi i (\sin\theta/\lambda) \hat{\mathbf{r}} \cdot \Pi(-\sin\theta, \cos\theta, 0)^T\right) d\hat{\mathbf{r}} \\ &= \int_{\mathcal{S}} Q(|\hat{\mathbf{r}}|) \exp\left(4\pi i (\sin\theta/\lambda) \hat{\mathbf{r}} \cdot (1,0,0)^T\right) d\hat{\mathbf{r}} \\ &= \int_{\mathcal{S}} Q(|\hat{\mathbf{r}}|) \exp\left(4\pi i (\sin\theta/\lambda) x_{\hat{\mathbf{r}}}\right) d\hat{\mathbf{r}}, \end{aligned} \tag{31}$$

where $x_{\hat{\mathbf{r}}}$ is the component of $\hat{\mathbf{r}}$ along the x axis. It is clear from (31) that f is a function of the ratio $\sin\theta/\lambda$; we write $f(\sin\theta/\lambda)$ to emphasize the dependence.

Experiments have determined the atomic scattering factors for many atoms as a function of $\sin\theta/\lambda$. For $\theta = 0$, the value of $f(\sin\theta/\lambda) = f(0)$ is simply equal to the number of electrons in the cloud, while the function decreases with as its argument increases.

By taking conjugates in (31), it is easy to see that $f(\sin\theta/\lambda)$ is real. In practice, however, deviations from symmetry give rise to nonzero imaginary components of $f(\sin\theta/\lambda)$.

We now consider the more interesting case in which the unit cell contains not a single atom but rather a molecule consisting of N atoms, located at positions

$$x_i \mathbf{a} + y_i \mathbf{b} + z_i \mathbf{c}, \quad i = 1, 2, \ldots, N,$$

where $\mathbf{a}$, $\mathbf{b}$, and $\mathbf{c}$ denote the basis vectors for the lattice and x_i, y_i and z_i are the coordinates of the center of the ith electron cloud relative to some reference point in the unit cell. Scattering takes place from each atomic cloud, and the effects of path length differences on the amplitude of scattering in each direction can be determined by similar arguments to those advanced above. Suppose we are given a direction $\mathbf{t}$ for the incident beam and a scattering direction $\mathbf{s}$ that satisfies (27) for some set of Miller indices (h, k, ℓ) (that is, $\mathbf{s}$ is a direction of strong scattering for the lattice structure formed by the crystal under investigation). We find that the phase difference between a ray scattered from the center of the ith electron cloud and one scattered from the molecular reference point is

$$(2\pi/\lambda)\lambda(x_i \mathbf{a} + y_i \mathbf{b} + z_i \mathbf{c}) \cdot (h\bar{\mathbf{a}} + k\bar{\mathbf{b}} + \ell\bar{\mathbf{c}}) = 2\pi(hx_i + ky_i + \ell z_i). \tag{32}$$

By applying this phase shift to the atomic scattering factor f_i associated with the ith atom, we obtain a contribution of

$$f_i \exp\left(2\pi(hx_i + ky_i + \ell z_i)\right)$$

to the scattering associated with the molecule, where f_i is given by (31). The total scattering in the direction characterized by (h, k, ℓ) and λ is then given by summing the contributions from each of the atoms, to obtain

$$F(h,k,\ell) \stackrel{\text{def}}{=} \sum_{i=1}^{N} f_i \exp\left(2\pi(hx_i + ky_i + \ell z_i)\right). \tag{33}$$

This quantity, a Fourier series, is known as the *structure factor*. In the absence of other factors that affect the scattering, the amplitude of the scattered ray in direction s will be $|F(h,k,\ell)|$.

Other factors such as temperature effects (which cause the molecules to oscillate around their mean position in the lattice) and polarization effects contribute to the intensity of each spot. The code LaueView aims to quantify each of these effects, so that the amplitudes of the structure factors can be recovered from the observed amplitude measurements.

Our discussions above about the interference patterns produced by the scattered rays assumed an infinite lattice. In finite lattices, the nonzero intensities can be detected also in directions that deviate slightly from the directions that satisfy (27). Microcrystals of unit cells fewer than 1000 in any dimension will give rise to non-Bragg scattering. The energy associated with each spot must be determined by integrating the intensity over a finite area on the detector that covers the spot in question. In typical applications of LaueView, the spots are often streaky because of mosaic spread of the crystal, and they often overlap, making it necessary to do a "deconvolution" to determine the intensity associated with each component spot. (The simple alternative approach of deleting overlapping spots from the data set degrades the quality of the data set considerably.)

Basic descriptions of X-ray diffraction appear in many books; we mention in particular those of Wilson [11] (from which some of the discussion of this section is drawn) and Glusker and Trueblood [3].

References

[1] I. J. Clifton, E. M. H. Duke, S. Wakatsuki, and Z. Ren. Evaluation of Laue diffraction patterns. *Methods in Enzymology*, 277:448–467, 1997.

[2] U. K. Genick, G. E. O. Borgstahl, K. Ng, Z. Ren, C. Pradervand, P. M. Burke, V. Srajer, T.-Y. Teng, W. Schildkamp, D. E. McRee, K. Moffatt, and E. Getzoff. Structure of a protein photocycle intermediate by millisecond time-resolved crystallography. *Science*, 275:1471–1475, March 1997.

[3] J. P. Glusker and K. N. Trueblood. *Crystal Structure Analysis: A Primer.* Oxford University Press, second edition, 1985.

[4] Keith Moffat. Laue diffraction. *Methods in Enzymology*, 277:433–447, 1997.

[5] Jorge J. Moré. The Levenberg-Marquardt algorithm: Implementation and theory. In G.A. Watson, editor, *Lecture Notes in Mathematics, No. 630–Numerical Analysis*, pages 105–116. Springer-Verlag, 1978.

[6] Jorge J. Moré and D.C. Sorensen. Computing a trust region step. *SIAM Journal on Scientific and Statistical Computing*, 4:553–572, 1983.

[7] W. H. Press, S. A. Teukolsky, and W. T. Vetterling. *Numerical Recipes in Fortran: The Art of Scientific Computing.* Cambridge University Press, second edition, 1992.

[8] Zhong Ren and Keith Moffat. Deconvolution of energy overlaps in Laue diffraction. *Journal of Applied Crystallography*, 28:482–493, 1995.

[9] Zhong Ren and Keith Moffat. Quantitative analysis of synchotron Laue diffraction patterns in macromolecular crystallography. *Journal of Applied Crystallography*, 28:461–481, 1995.

[10] Zhong Ren, Kingman Ng, Gloria E. O. Borgstahl, Elizabeth D. Getzoff, and Keith Moffat. Quantitative analysis of time-resolved Laue diffraction patterns. *Journal of Applied Crystallography*, 29:246–260, 1996.

[11] A. J. C. Wilson. *Elements of X-Ray Crystallography.* Addison-Wesley, Reading, Mass., 1970.